

Continuous multilinguality with language vectors

Robert Östling

Department of Modern Languages
University of Helsinki

Jörg Tiedemann

Department of Modern Languages
University of Helsinki

Abstract

Most existing models for multilingual natural language processing (NLP) treat language as a discrete category, and make predictions for either one language or the other. In contrast, we propose using continuous vector representations of language. We show that these can be learned efficiently with a character-based neural language model, and used to improve inference about language varieties not seen during training. In experiments with 1303 Bible translations into 990 different languages, we empirically explore the capacity of multilingual language models, and also show that the language vectors capture genetic relationships between languages.

1 Introduction

Neural language models (Bengio et al., 2003; Mikolov et al., 2010; Sundermeyer et al., 2012) have become an essential component in several areas of natural language processing (NLP), such as machine translation, speech recognition and image captioning. They have also become a common benchmarking application in machine learning research on recurrent neural networks (RNN), because producing an accurate probabilistic model of human language is a very challenging task which requires all levels of linguistic analysis, from pragmatics to phonology, to be taken into account.

A typical language model is trained on text in a single language, and if one needs to model multiple languages the standard solution is to train a separate model for each language. This presupposes large quantities of monolingual data in each of the languages that needs to be covered and each

model with its parameters is completely independent of any of the other models.

We propose instead to use a single model with real-valued vectors to indicate the language used, and to train this model with a large number of languages. We thus get a language model whose predictive distribution $p(x_t|x_{1..t-1}, l)$ is a continuous function of the language vector l , a property that is trivially extended to other neural NLP models. In this paper, we explore the “language space” containing these vectors, and in particular explore what happens when we move beyond the points representing the languages of the training corpus.

In contrast to related work, we focus on massively multilingual data sets to cover for the first time a substantial amount of the linguistic diversity in the world in a project related to data-driven language modeling. We do not presuppose any prior knowledge about language similarities and evolution and let the model discover relations on its own purely by looking at the data. The only supervision that is giving during training is a language identifier as a one-hot encoding. From that and the actual training examples, the system learns dense vector representations for each language included in our data set along with the character-level RNN parameters of the language model itself.

2 Related Work

Concurrent with this work, Johnson et al. (2016) conducted a study using neural machine translation (NMT), where a sub-word decoder is told which language to generate by means of a special language identifier token in the source sentence. This is close to our model, although beyond a simple interpolation experiment (as in our Section 5.3) they did not further explore the language vectors, which would have been challenging to do given

the small number of languages used in their study.

Ammar et al. (2016) used one-hot language identifiers as input to a multilingual word-based dependency parser, based on multilingual word embeddings. Given that they report this resulting in higher accuracy than using features from a typological database, it is a reasonable guess that their system learned language vectors which were able to encode syntactic properties relevant to the task. Unfortunately, they also did not look closer at the language vector space, which would have been interesting given the relatively large and diverse sample of languages represented in the Universal Dependencies treebanks.

3 Data

We base our experiments on a large collection of Bible translations coming from various sources and periods of times. Any other multilingual data collection would work as well, but with the selected corpus we have the advantage that we cover the same genre and roughly the same coverage for each language involved. It is also easy to divide the data into training and test sets by using Bible verse numbers, which allows us to control for semantic similarity between languages in a way that would have been difficult in a corpus that is not multi-parallel. Altogether we have 1,303 translations in 990 languages that we can use for our purposes. These were chosen so that the model alphabet size is below 1000 symbols, which was satisfied by choosing only translations in Latin, Cyrillic or Greek script.

The corpus deviates in some ways from an ideal multi-parallel corpus. Most translations are of the complete New Testament, whereas around 300 also contain the Old Testament (thus several times longer), and around ten contain only portions of the New Testament. Additionally, several languages have multiple translations, which are then concatenated. These translations may vary in age and style, but historical versions of languages (with their own ISO 639-3 code) are treated as distinct languages. During training we enforce a uniform distribution between languages when selecting training examples.

4 Methods

Our model is based on a standard stacked character-based LSTM (Hochreiter and Schmidhuber, 1997) with two layers, followed by a hid-

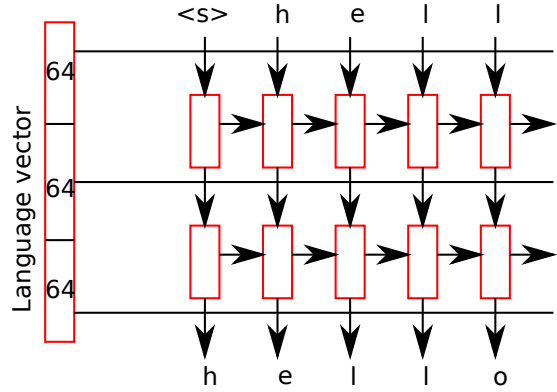


Figure 1: Schematic of our model. The three parts of the language vector are concatenated with the inputs to the two LSTM:s and the final softmax layer.

den layer and a final output layer with softmax activations. The only modification made to accommodate the fact that we train the model with text in nearly a thousand languages, rather than one, is that language embedding vectors are concatenated to the inputs of the LSTMs at each time step and the hidden layer before the softmax. We used three separate embeddings for these levels, in an attempt to capture different types of information about languages. The model structure is summarized in Figure 1.

In our experiments we use 1024-dimensional LSTMs, 128-dimensional character embeddings, and 64-dimensional language embeddings. Layer normalization (Ba et al., 2016) is used, but no dropout or other regularization since the amount of data is very large (about 3 billion characters) and training examples are seen at most twice. For smaller models early stopping is used. We use Adam (Kingma and Ba, 2014) for optimization. Training takes between an hour and a few days on a K40 GPU, depending on the data size.

5 Results

In this section, we present several experiments with the model described. For exploring the language vector space, we use hierarchical agglomerative clustering for visualization. For measuring performance, we use cross-entropy on held out-data. For this, we use a set of the 128 most commonly translated Bible verses, to ensure that the held-out set is as large and overlapping as possible among languages.

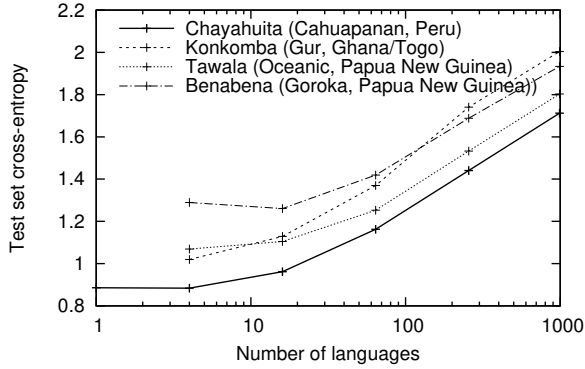


Figure 2: Cross-entropy of the test sets from the first four languages added to our model. At the leftmost point ($x = 1$), *only* Chayahuita is used for training the model so no results are available for the other languages.

5.1 Model capacity

Our first experiment tries to answer what happens when more and more languages are added to the model. There are two settings: adding languages in a random order, or adding the most closely related languages first. Cross-entropy plots for these settings are shown in Figure 2 and Figure 3.

In both cases, the model degrades gracefully (or even improves) for a number of languages, but then degrades linearly (i.e. exponential growth of perplexity) with exponentially increasing number of languages.

For comparison, Figure 3 compares this to the effect of decreasing the number of parameters in the LSTM. Here the behavior is similar, but unlike the Swedish model which got somewhat better when closely related languages were added, the increase in cross-entropy is monotone.

5.2 Structure of the language space

We now take a look at the language vectors found during training with the full model of 990 languages. Figure 4 shows a hierarchical clustering of the subset of Germanic languages, which closely matches the established genetic relationships in this language family. While our experiments indicate that finding more remote relationships (say, connecting the Germanic languages to the Celtic) is difficult for the model, it is clear that the language vectors preserves similarity properties between languages.

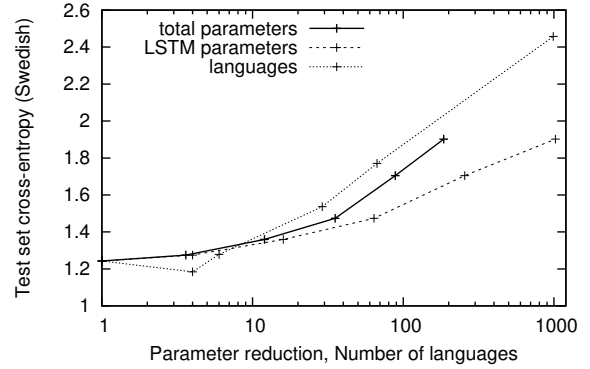


Figure 3: Cross-entropy of the Swedish test set, given two conditions: increasing number of languages (adding the most similar languages first) or decreasing number of parameters (for a monolingual model, which is why the curves meet at $x = 1$). Curves are given both when measuring the total number of model parameters or only the LSTM parameters, because the former way of measuring is more standard but the latter makes more sense in our case.

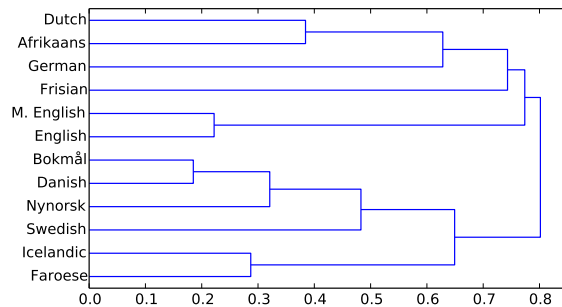


Figure 4: Hierarchical clustering of language vectors of Germanic languages.

5.3 Generating Text

Since our language model is conditioned on a language vector, we can gain some intuitive understanding of the language space by generating text from different points in it. These points could be either one of the vectors learned during training, or some arbitrary other point. Table 1 shows text samples from different points along the line between Modern English [*eng*] and Middle English [*enm*]. Consistent with the results of Johnson et al. (2016), it appears that the interesting region lies rather close to 0.5. Compare also to our Figure 5, which shows that up until about a third of the way between English and German, the language model is nearly perfectly tuned to English.

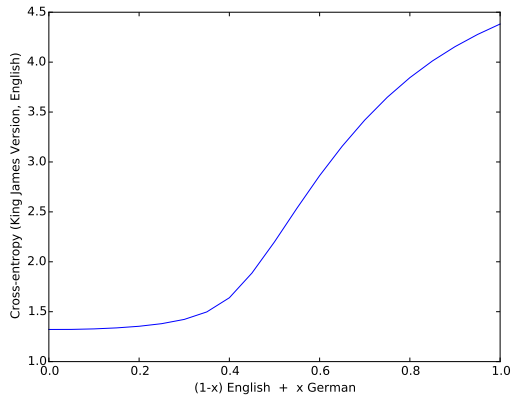


Figure 5: Cross-entropy of interpolated language models for English and German measured on English held-out text.

5.4 Mixing and Interpolating Between Languages

By means of cross-entropy, we can also visualize the relation between languages in the multilingual space. Figure 5 plots the interpolation results for two relatively dissimilar languages, English and German. As expected, once the language vector moves too close to the German one, model performance drops drastically.

More interesting results can be obtained if we interpolate between two language variants and compute cross-entropy of a text that represents an intermediate form. Figure 6 shows the cross-entropy of the King James Version of the Bible (published 1611), when interpolating between Modern English (1500–) and Middle English (1050–1500). The optimal point turns out to be close to the midway point between them.

5.5 Language identification

If we have a sample of an unknown language or language variant, it is possible to estimate its language vector by backpropagating through the language model with all parameters except the language vector fixed. We found that a very small set of sentences is enough to give a considerable improvement in cross-entropy on held-out sentences. In this experiment, we used 32 sentences from the King James Version of the Bible. Using the resulting language vector, test set cross-entropy improved from 1.39 (using the Modern English language vector as initial value) to 1.35. This is comparable to the result obtained in Section 5.4, except that here we do not restrict the search space

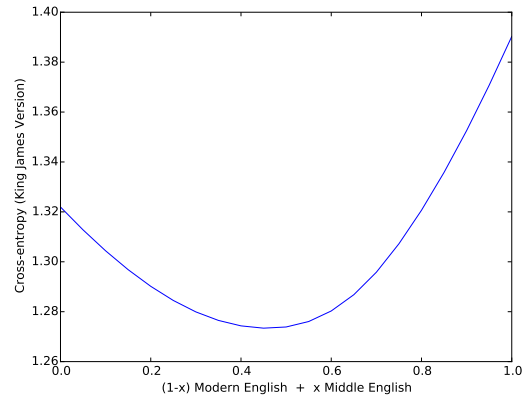


Figure 6: Cross-entropy of interpolated language models for modern and middle English tested on data from the King James Bible.

Table 1: Examples generated by interpolating between Modern English and Middle English.

%	Random sample (temperature parameter $\tau = 0.5$)
30	and thei schulen go in to alle these thingis, and schalt endure bothe in the weie
40	and there was a certaine other person who was called in a dreame that he went into a mountaine.
48	and god sayd, i am the light of the world, and the powers of the enemies of the most high god may find first for many.
50	but if there be some of the seruants, and to all the people, and the angels of god, and the prophets
52	then he came to the gate of the city, and the bread was to be brought
60	and the man whom the son of man is born of god, so have i therefore already sent to the good news of christ.

to points on a straight line between two language vectors.

6 Conclusions

We have shown that *language vectors* can be learned efficiently from raw text and possess several interesting properties. First, they capture language similarity to the extent that language family trees can be reconstructed by clustering the vectors. Second, they allow us to interpolate between languages in a sensible way, and even allow adopting the model using a very small set of text, simply by optimizing the language vector.

References

- [Ammar et al.2016] Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.
- [Ba et al.2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv e-prints*, July.
- [Bengio et al.2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, March.
- [Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780. doi: 10.1162/neco.1997.9.8.1735.
- [Johnson et al.2016] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google’s multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558.
- [Kingma and Ba2014] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.
- [Mikolov et al.2010] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, volume 2010, pages 1045–1048. International Speech Communication Association.
- [Sundermeyer et al.2012] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, pages 194–197.